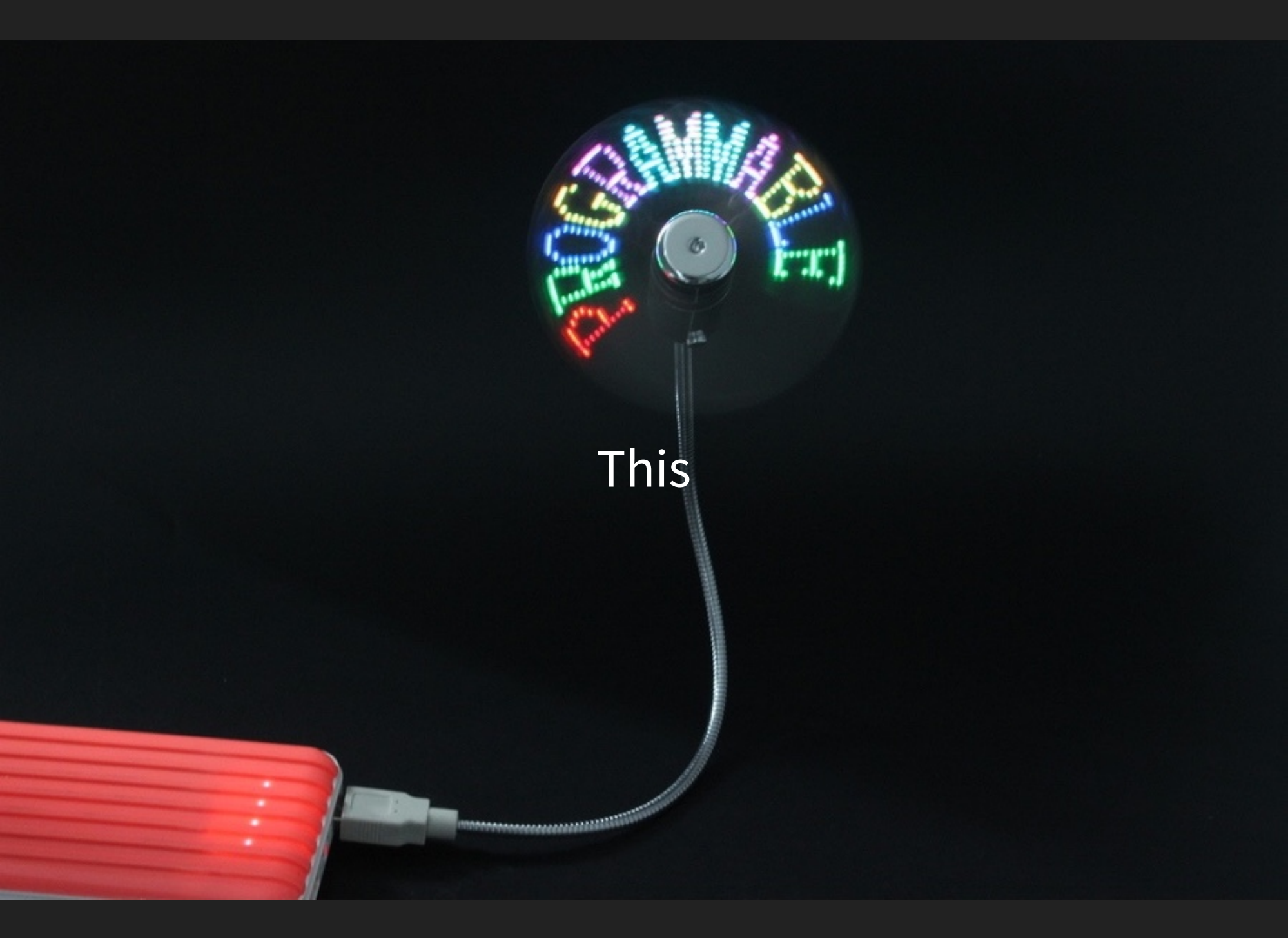


UNDERSTANDING MICRO WAVE

Fergus Symon





This

WHY?

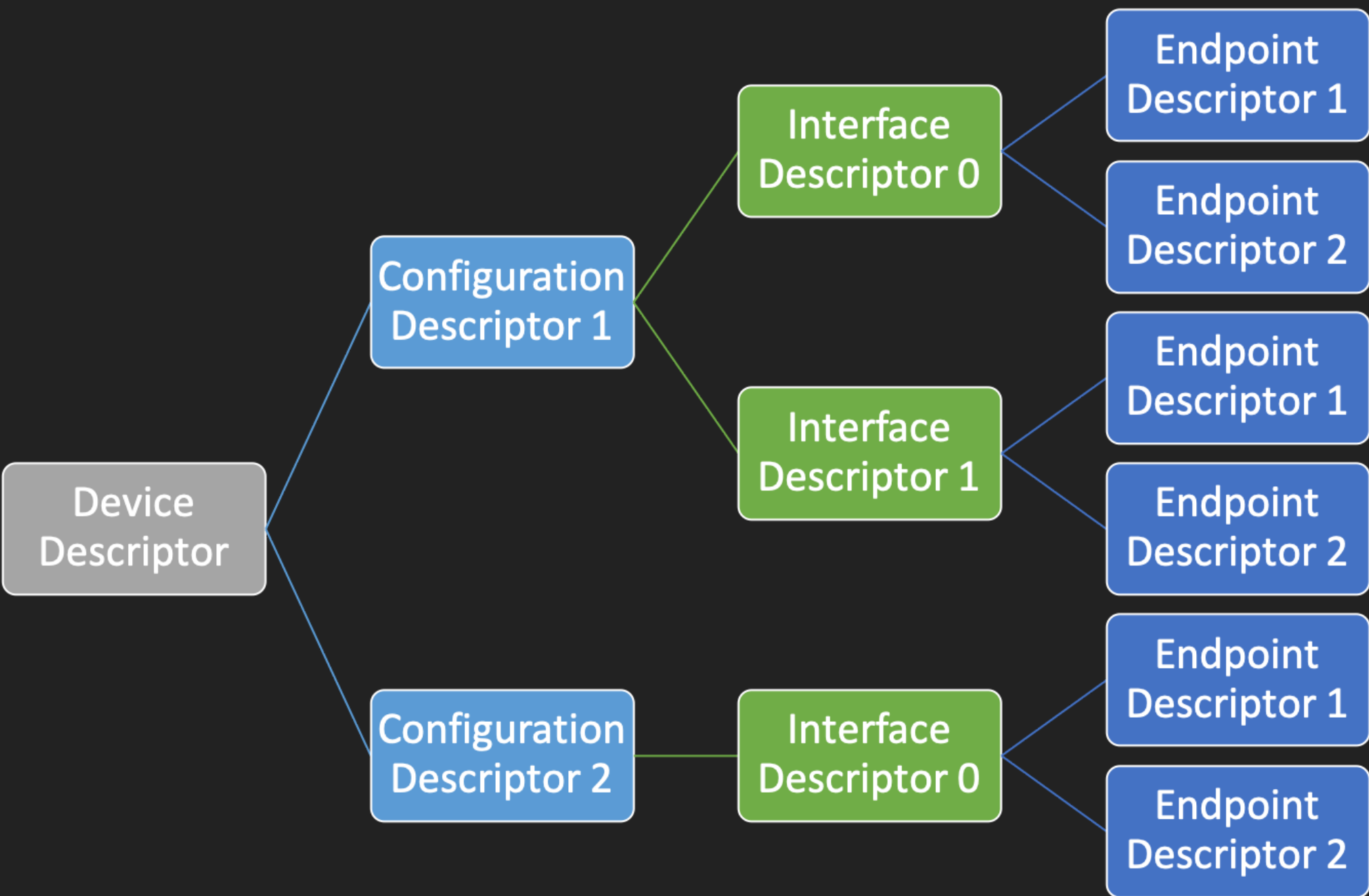
I wanted a system for home notifications

- RGB LED Display
- "Computer Programmed" (USB)
- No Driver



UNIVERSAL SERIAL BUS

- Four pins
 - One power (5V)
 - One ground
 - Two transmission (differential pair)
- Brilliant versioning scheme
- Highly standardised



- Transactions
- Pipes
 - Control: simple messages
 - Bulk: maximum bandwidth, no guarantees on bandwidth or latency
 - Isochronous: guaranteed bandwidth (best effort)
 - Interrupt: low latency



Back Here


"No Driver" is an interesting claim 🤔

- Can't be a vendor specific device class... right?
- Is there a communications device that has drivers on Windows?
- Is it a mass storage device and use file transfers to operate?
- Can something act as a HID device and just use custom control messages?

Almost! 🤨

REVERSING USB PROTOCOLS

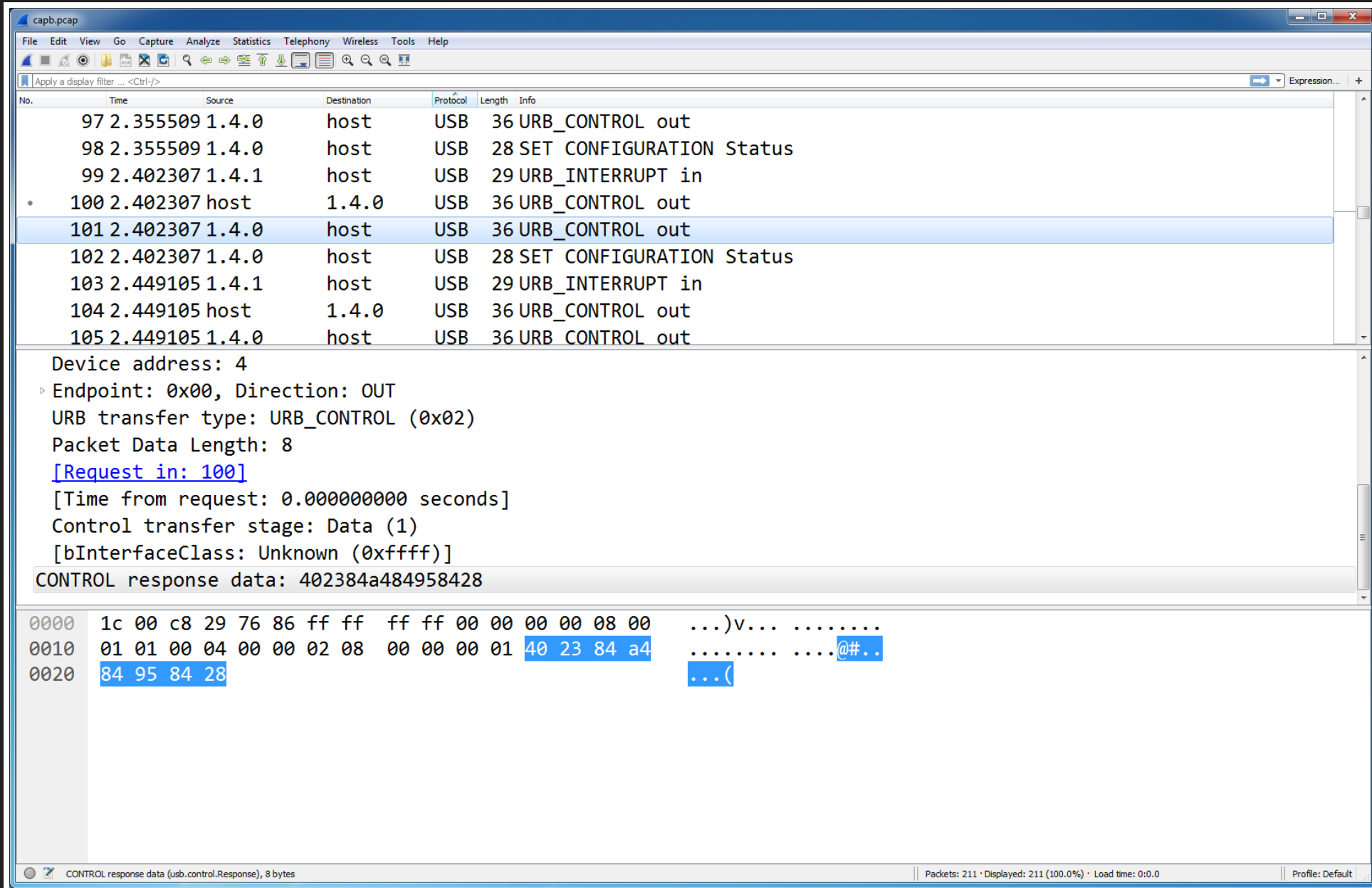
Two approaches:

1. Black-box analysis: capture USB messages
determine differences
2. White-box analysis: reverse-engineer the software
on one of the end-points
3. ... or both 

Capturing USB messages: 💰 mode



Capturing USB messages: 🙄 mode



capb.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
97	2.355509	1.4.0	host	USB	36	URB_CONTROL out
98	2.355509	1.4.0	host	USB	28	SET CONFIGURATION Status
99	2.402307	1.4.1	host	USB	29	URB_INTERRUPT in
• 100	2.402307	host	1.4.0	USB	36	URB_CONTROL out
101	2.402307	1.4.0	host	USB	36	URB_CONTROL out
102	2.402307	1.4.0	host	USB	28	SET CONFIGURATION Status
103	2.449105	1.4.1	host	USB	29	URB_INTERRUPT in
104	2.449105	host	1.4.0	USB	36	URB_CONTROL out
105	2.449105	1.4.0	host	USB	36	URB_CONTROL out

Device address: 4

Endpoint: 0x00, Direction: OUT

URB transfer type: URB_CONTROL (0x02)

Packet Data Length: 8

[\[Request in: 100\]](#)

[Time from request: 0.000000000 seconds]

Control transfer stage: Data (1)

[bInterfaceClass: Unknown (0xffff)]

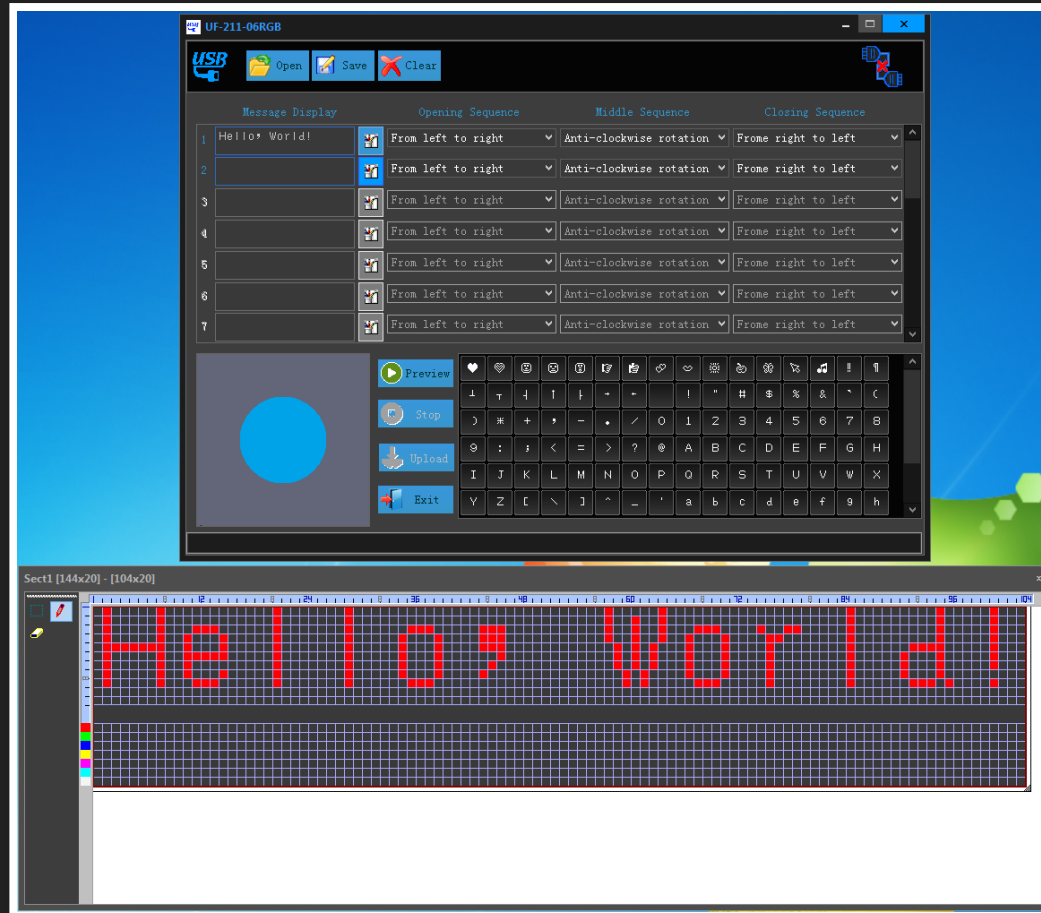
CONTROL response data: 402384a484958428

```
0000  1c 00 c8 29 76 86 ff ff ff ff 00 00 00 08 00  ... )v... ..
0010  01 01 00 04 00 00 02 08 00 00 00 01 40 23 84 a4  ..... @#..
0020  84 95 84 28  ... (
```

CONTROL response data (usb.control.Response), 8 bytes

Packets: 211 · Displayed: 211 (100.0%) · Load time: 0:0.0

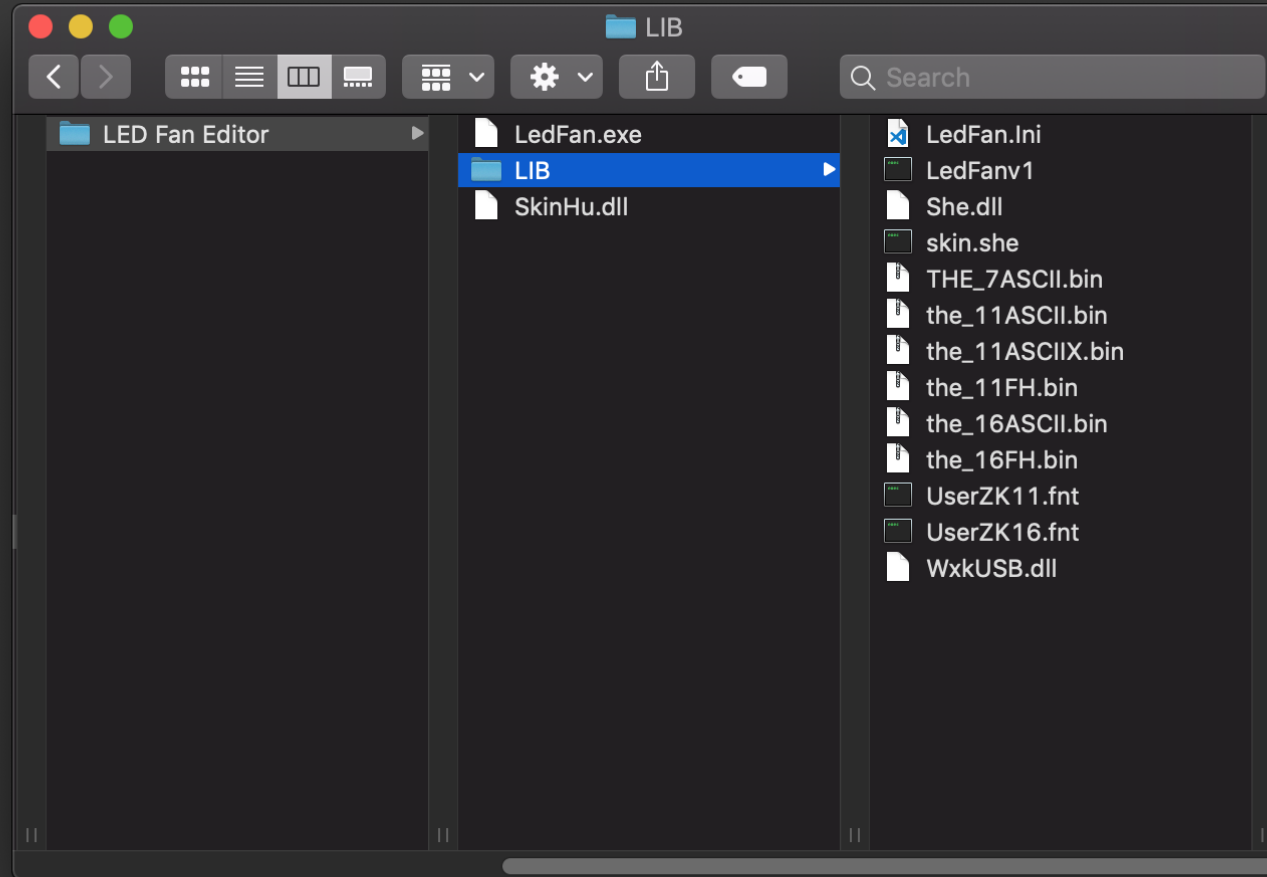
Profile: Default



Dynamic analysis alone is useless



What else can we find?



WxkUSB.dll

File Edit Jump Search View Debugger Options Windows Help

Library function Regular function Instruction Data Unexplored External symbol

Functions window

Function name

- DllMain(x,x,x)
- OpenUSBDevice
- CloseUSBDevice
- WriteUSB
- ReadUSB
- GetInputLength
- GetOutputLength
- GetErrorMsgA
- GetErrorMsgW
- SetOutputReport
- GetInputReport
- HidD_GetAttributes
- HidD_GetHidGuid
- HidD_FreePreparedData

Graph overview

IDA View-A Hex View-1 Structures Enums Imports Exports

Name	Address	Ordinal
OpenUSBDevice	0000000010001010	1
CloseUSBDevice	0000000010001170	2
ReadUSB	0000000010001210	3
WriteUSB	0000000010001180	4
GetInputLength	00000000100012A0	5
GetOutputLength	00000000100012F0	6
GetErrorMsgA	0000000010001340	7
GetErrorMsgW	0000000010001370	8
SetOutputReport	00000000100013A0	9
GetInputReport	00000000100013C0	10
DllEntryPoint	0000000010001706	[main entry]

Line 1 of 11

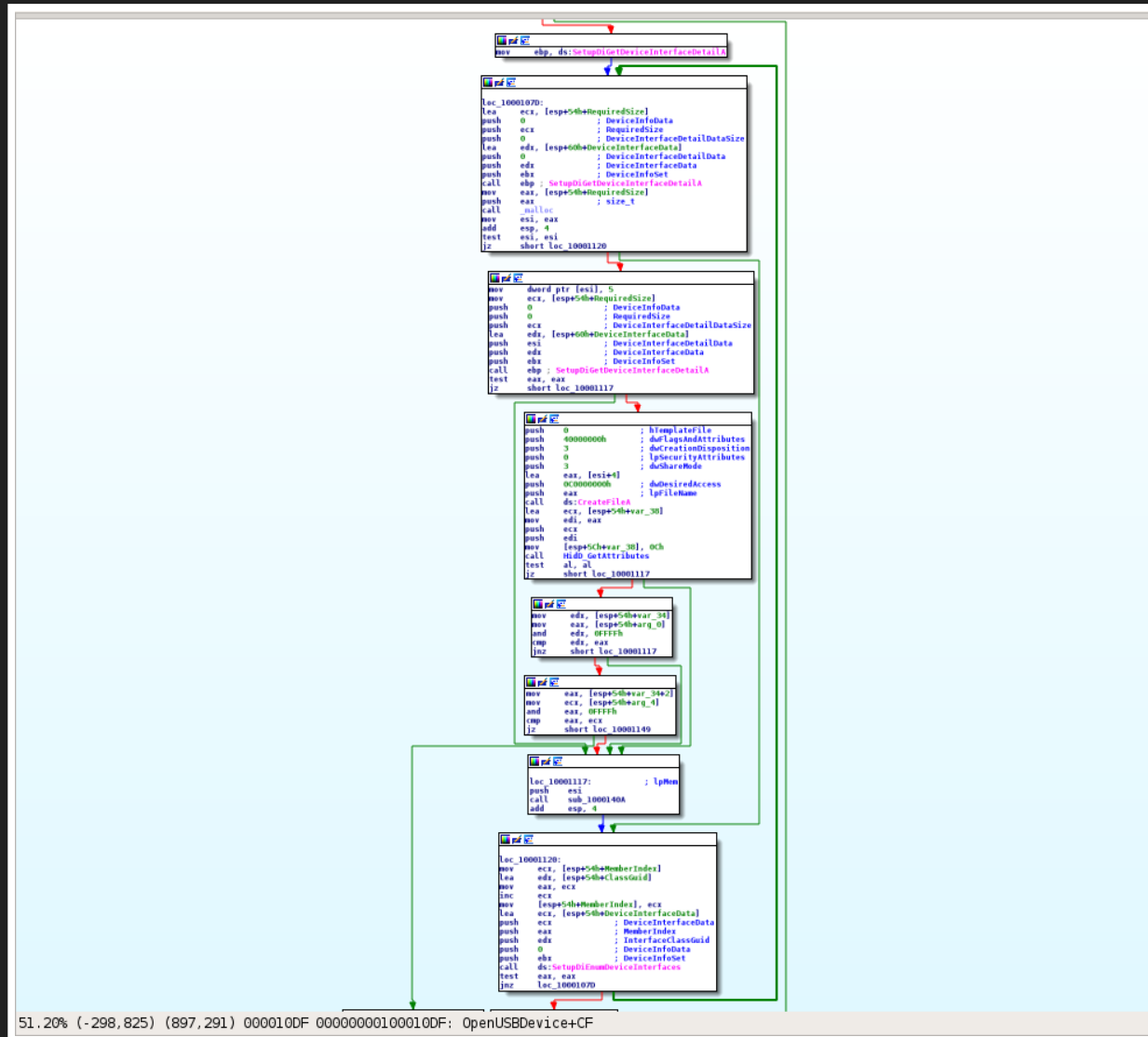
Output window

The initial autoanalysis has been finished.

Python

AU: idle Down Disk: 11GB

OpenUSBDevice(WORD, WORD)



WriteUSB(HANDLE, LPCVOID, DWORD, LPDWORD)

```
; Exported entry 4. WriteUSB

; int __stdcall WriteUSB(HANDLE hFile, LPCVOID lpBuffer, DWORD nNumberOfBytesToWrite, LPDWORD lpNumberOfBytesWritten)
public WriteUSB
WriteUSB proc near

    Overlapped= _OVERLAPPED ptr -14h
    hFile= dword ptr 4
    lpBuffer= dword ptr 8
    nNumberOfBytesToWrite= dword ptr 0Ch
    lpNumberOfBytesWritten= dword ptr 10h

    sub     esp, 14h
    xor     eax, eax
    mov     edx, [esp+14h+nNumberOfBytesToWrite]
    mov     [esp+14h+Overlapped.Internal], eax
    push    esi
    mov     esi, [esp+18h+lpNumberOfBytesWritten]
    mov     [esp+18h+Overlapped.InternalHigh], eax
    mov     [esp+18h+Overlapped.Offset], eax
    lea     ecx, [esp+18h+Overlapped]
    mov     [esp+18h+Overlapped.OffsetHigh], eax
    push    edi
    mov     edi, [esp+1Ch+hFile]
    mov     [esp+1Ch+Overlapped.hEvent], eax
    mov     [esp+1Ch+Overlapped.Offset], eax
    mov     [esp+1Ch+Overlapped.OffsetHigh], eax
    mov     [esp+1Ch+Overlapped.hEvent], eax
    mov     eax, [esp+1Ch+lpBuffer]
    push    ecx           ; lpOverlapped
    push    esi           ; lpNumberOfBytesWritten
    push    edx           ; nNumberOfBytesToWrite
    push    eax           ; lpBuffer
    push    edi           ; hFile
    call    ds:WriteFile
    test    eax, eax
    jz      short loc_100011D7
```

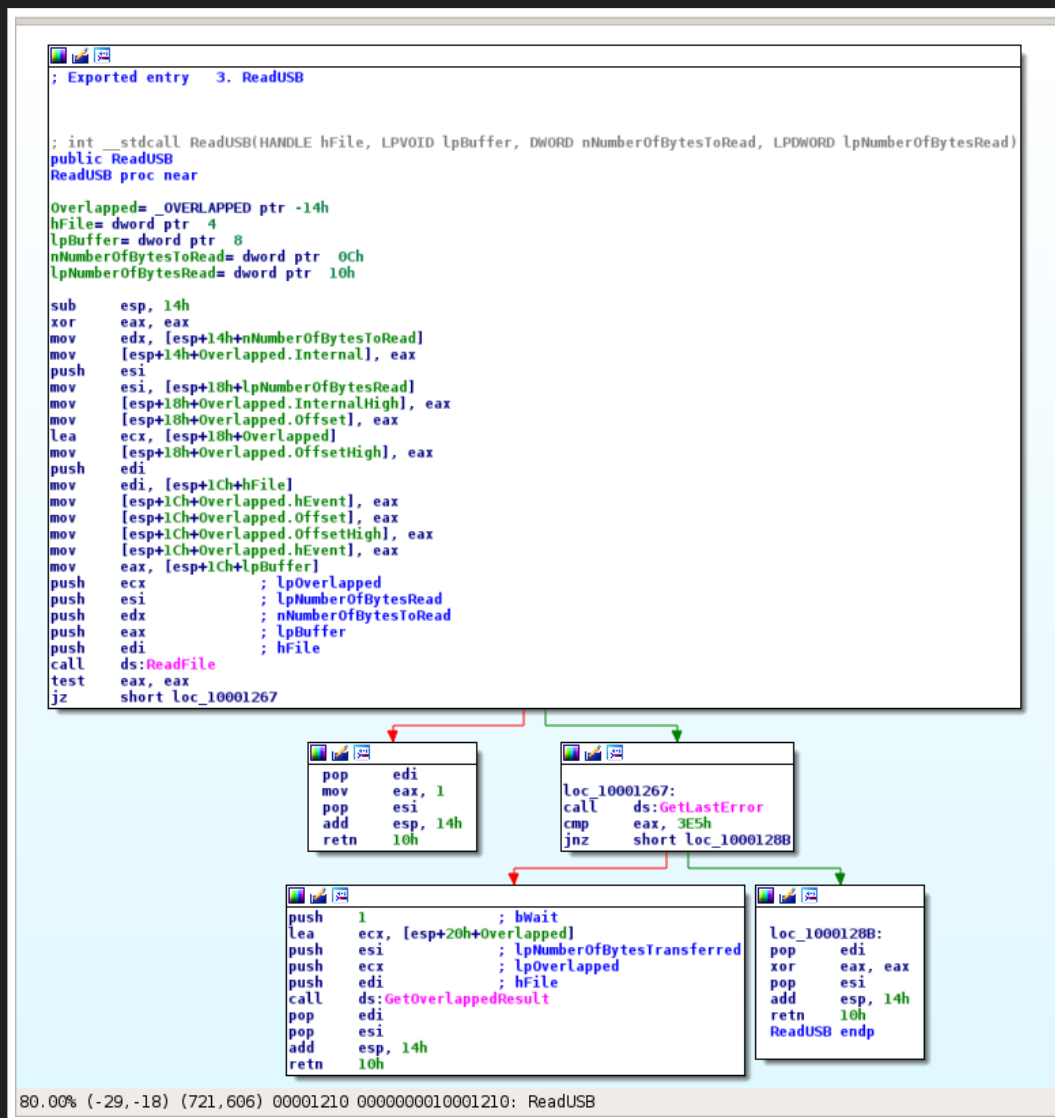
```
pop     edi
mov     eax, 1
pop     esi
add     esp, 14h
retn    10h
```

```
loc_100011D7:
call    ds:GetLastError
cmp     eax, 3E5h
jnz     short loc_100011FB
```

```
push    1                ; bWait
lea     ecx, [esp+20h+Overlapped]
push    esi              ; lpNumberOfBytesTransferred
push    ecx              ; lpOverlapped
push    edi              ; hFile
call    ds:GetOverlappedResult
pop     edi
pop     esi
add     esp, 14h
retn    10h
```

```
loc_100011FB:
pop     edi
xor     eax, eax
pop     esi
add     esp, 14h
retn    10h
WriteUSB endp
```

ReadUSB(HANDLE, LPCVOID, DWORD, LPDWORD)



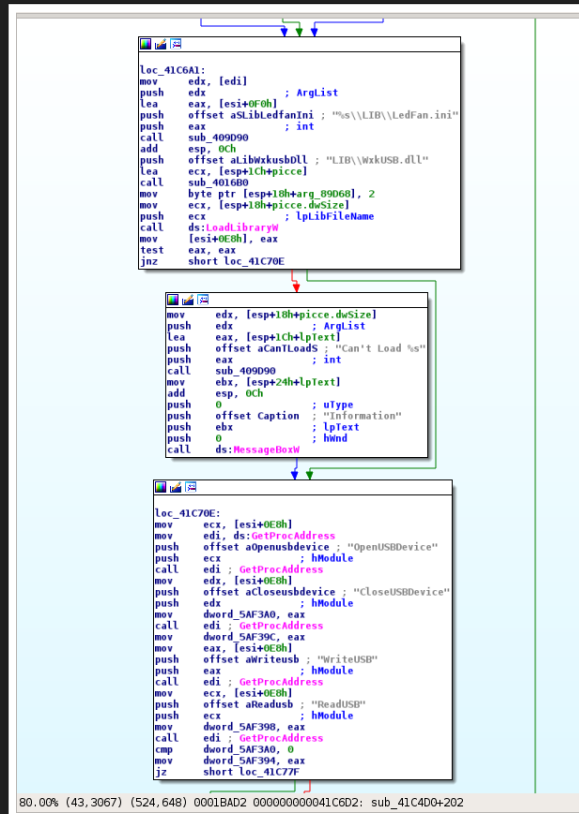
This has been great!

We know it's using HID reports, but nothing about the protocol 😞

What uses WxkUSB.dll?

Looking at the import tables of the other binaries:
nothing 😞

Hold up, I know this game... 🤔



```
loc_41C6A1:
mov     edx, [edi]
push    edx
lea     eax, [esi+0F0h]
push    offset aLibLefanIni ; ""s\\LIB\\Lefan.ini"
push    eax
call    sub_409D90
add     esp, 0Ch
lea     ecx, [esp+1Ch+picce]
call    sub_4016B0
mov     byte ptr [esp+18h+arg_89066], 2
mov     ecx, [esp+18h+picce.dwSize]
push    ecx
call    ds:LoadLibraryW
mov     [esi+0E8h], eax
test    eax, eax
jnz     short loc_41C70E

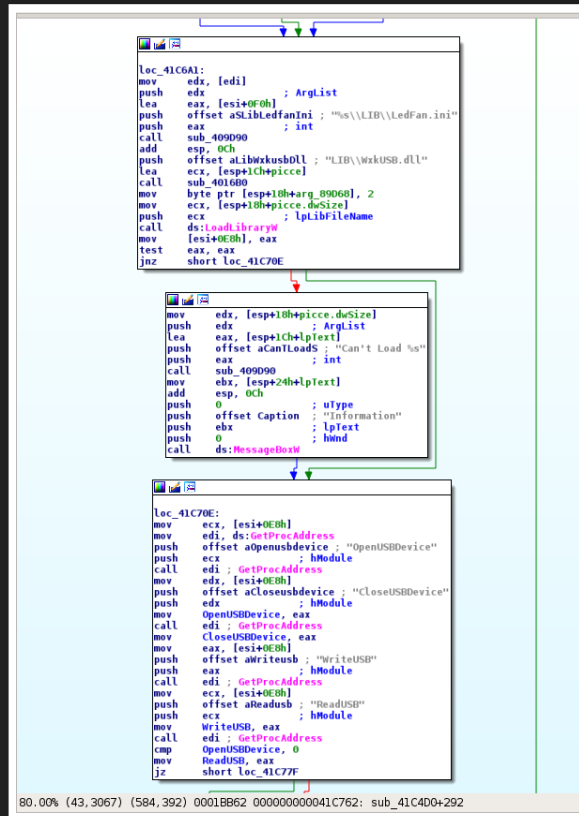
mov     edx, [esp+18h+picce.dwSize]
push    edx
lea     eax, [esp+1Ch+lpText]
push    offset aCanLoadS ; "Can't Load %s"
push    eax
call    sub_409D90
mov     ebx, [esp+24h+lpText]
add     esp, 0Ch
push    0
push    offset Caption ; "Information"
push    ebx
push    0
call    ds:MessageBoxW

loc_41C70E:
mov     ecx, [esi+0E8h]
mov     edi, ds:GetProcAddress
push    offset aOpenusbdevice ; "OpenUSBDevice"
push    ecx
call    edi
mov     edi, [esi+0E8h]
push    offset aCloseusbdevice ; "CloseUSBDevice"
push    ecx
call    edi
mov     dword_5AF3A0, eax
call    ds:GetProcAddress
mov     eax, [esi+0E8h]
push    offset aWriteusb ; "WriteUSB"
push    eax
call    ds:GetProcAddress
mov     ecx, [esi+0E8h]
push    offset aReadusb ; "ReadUSB"
push    ecx
call    ds:GetProcAddress
mov     dword_5AF3A0, eax
cmp     dword_5AF3A0, 0
mov     dword_5AF394, eax
jz      short loc_41C77F

80.00% (43,3067) (524,648) 0001BAD2 000000000041C6D2: sub_41C4D0+202
```

Just quickly clean this up...

Hold up, I know this game... 🤔



```
loc_41C6A1:
mov     edx, [edi]
push    edx                ; ArgList
lea     eax, [esi+0F0h]
push    offset aSlibLefanini ; ""s\\LIB\\Lefan.ini"
push    eax                ; int
call    sub_409D90
add     esp, 0Ch
push    offset aLibWxusbDll ; "LIB\\Wxusb.dll"
lea     ecx, [esp+1Chpicce]
sub     sub_4016B0
call    byte ptr [esp+18h+arg_09D60], 2
mov     ecx, [esp+18h+picce_dsSize]
push    ecx                ; lpLibFileName
call    ds:LoadLibraryW
mov     [esi+0E8h], eax
test    eax, eax
jnz     short loc_41C70E

mov     edx, [esp+18h+picce_dsSize]
push    edx                ; ArgList
lea     eax, [esp+1Ch+lpText]
push    offset aCanLoadS ; "Can't Load %s"
push    eax                ; int
call    sub_409D90
mov     ebx, [esp+24h+lpText]
add     esp, 0Ch
push    0                  ; uType
push    offset Caption ; "Information"
push    ebx                ; lpText
push    0                  ; hwnd
call    ds:MessageBoxW

loc_41C70E:
mov     ecx, [esi+0E8h]
mov     edi, ds:GetProcAddress ; "OpenUSBDevice"
push    offset aOpenusbdevice ; "OpenUSBDevice"
push    ecx                ; hModule
call    edi ; GetProcAddress
mov     edx, [esi+0E8h]
push    offset aCloseusbdevice ; "CloseUSBDevice"
push    edx                ; hModule
call    OpenUSBDevice, eax
call    edi ; GetProcAddress
CloseUSBDevice, eax
mov     eax, [esi+0E8h]
push    offset aWriteusb ; "WriteUSB"
push    eax                ; hModule
call    edi ; GetProcAddress
mov     ecx, [esi+0E8h]
push    offset aReadusb ; "ReadUSB"
push    ecx                ; hModule
mov     WriteUSB, eax
call    edi ; GetProcAddress
cmp     OpenUSBDevice, 0
mov     ReadUSB, eax
jz      short loc_41C77F

80.00% (43,3067) (584,392) 0001BB62 000000000041C762: sub_41C4D0+292
```

Just quickly clean this up...

```

loc_41C6A1:
mov     edx, [edi]
push    edx                ; ArgList
lea     eax, [esi+0F0h]
push    offset aSLibLedfanIni ; "%s\\LIB\\LedFan.ini"
push    eax                ; int
call    sub_409D90
add     esp, 0Ch
push    offset aLibWxusbDll ; "LIB\\WxkUSB.dll"
lea     ecx, [esp+1Ch+picce]
call    sub_4016B0
mov     byte ptr [esp+18h+arg_89D68], 2
mov     ecx, [esp+18h+picce.dwSize]
push    ecx                ; lpLibFileName
call    ds:LoadLibraryW
mov     [esi+0E8h], eax
test    eax, eax
jnz     short loc_41C70E

```

```

mov     edx, [esp+18h+picce.dwSize]
push    edx                ; ArgList
lea     eax, [esp+1Ch+lpText]
push    offset aCantLoadS ; "Can't Load %s"
push    eax                ; int
call    sub_409D90
mov     ebx, [esp+24h+lpText]
add     esp, 0Ch
push    0                  ; uType
push    offset Caption ; "Information"
push    ebx                ; lpText
push    0                  ; hWnd
call    ds:MessageBoxW

```

```

loc_41C70E:
mov     ecx, [esi+0E8h]
mov     edi, ds:GetProcAddress
push    offset aOpenusbdevice ; "OpenUSBDevice"
push    ecx                ; hModule
call    edi ; GetProcAddress
mov     edx, [esi+0E8h]
push    offset aCloseusbdevice ; "CloseUSBDevice"
push    edx                ; hModule
mov     OpenUSBDevice, eax
call    edi ; GetProcAddress
mov     CloseUSBDevice, eax
mov     eax, [esi+0E8h]
push    offset aWriteusb ; "WriteUSB"
push    eax                ; hModule
call    edi ; GetProcAddress
mov     ecx, [esi+0E8h]
push    offset aReadusb ; "ReadUSB"
push    ecx                ; hModule
mov     WriteUSB, eax
call    edi ; GetProcAddress
cmp     OpenUSBDevice, 0
mov     ReadUSB, eax
jz      short loc_41C77F

```

LedFan.exe: 2.4MB of GUI fun!

File Edit Jump Search View Debugger Options Windows Help

Library function Regular function Instruction Data Unexplored External symbol

Functions window

Function name

- sub_401000
- sub_401010
- sub_401020
- nullsub_4
- sub_401040
- sub_401050
- sub_401080
- sub_4010A0
- sub_4010D0
- sub_401140
- sub_401180
- sub_401210
- sub_4012E0
- sub_401310

Line 3 of 8983

Graph overview

IDA View-A Hex View-1 Structures Enums Imports Exports

Attributes: library function bp-based frame

```
; int __stdcall wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPWSTR lpCmdLine, int nShowCmd)
_wWinMain@16 proc near

hInstance= dword ptr 8
hPrevInstance= dword ptr 0Ch
lpCmdLine= dword ptr 10h
nShowCmd= dword ptr 14h

mov     edi, edi
push    ebp
mov     ebp, esp
pop     ebp
jmp     loc_5419BC
```

loc_541992:

```
mov     edi, edi
push    ebp
mov     ebp, esp
call    ?AfxGetModuleState@@YGPA
mov     cl, byte ptr [ebp+hInsta
mov     [eax+14h], cl
xor     eax, eax
inc     eax
pop     ebp
ret     8
```

loc_5419BC:

```
mov     edi, edi
push    ebp
mov     ebp, esp
push    ebx
push    esi
push    edi
or      ebx, 0FFFFFFFh
call    ?AfxGetThread@@YGPAVCWinThread@@@XZ ; AfxGetThread(void)
mov     esi, eax
call    ?AfxGetModuleState@@YGPAVAFX_MODULE_STATE@@@XZ ; AfxGetModuleState(void)
push    [ebp+nShowCmd] ; int
mov     edi, [eax+4]
push    [ebp+lpCmdLine] ; wchar_t *
push    [ebp+hPrevInstance] ; HINSTANCE
push    [ebp+hInstance] ; HINSTANCE
```

100.00% (-46, -21) (28, 63) 00140D87 00000000000541987: wWinMain(x,x,x,x) (Synchronized with Hex View-1)

Output window

The initial autoanalysis has been finished.

Python

AU: idle Down Disk: 11GB

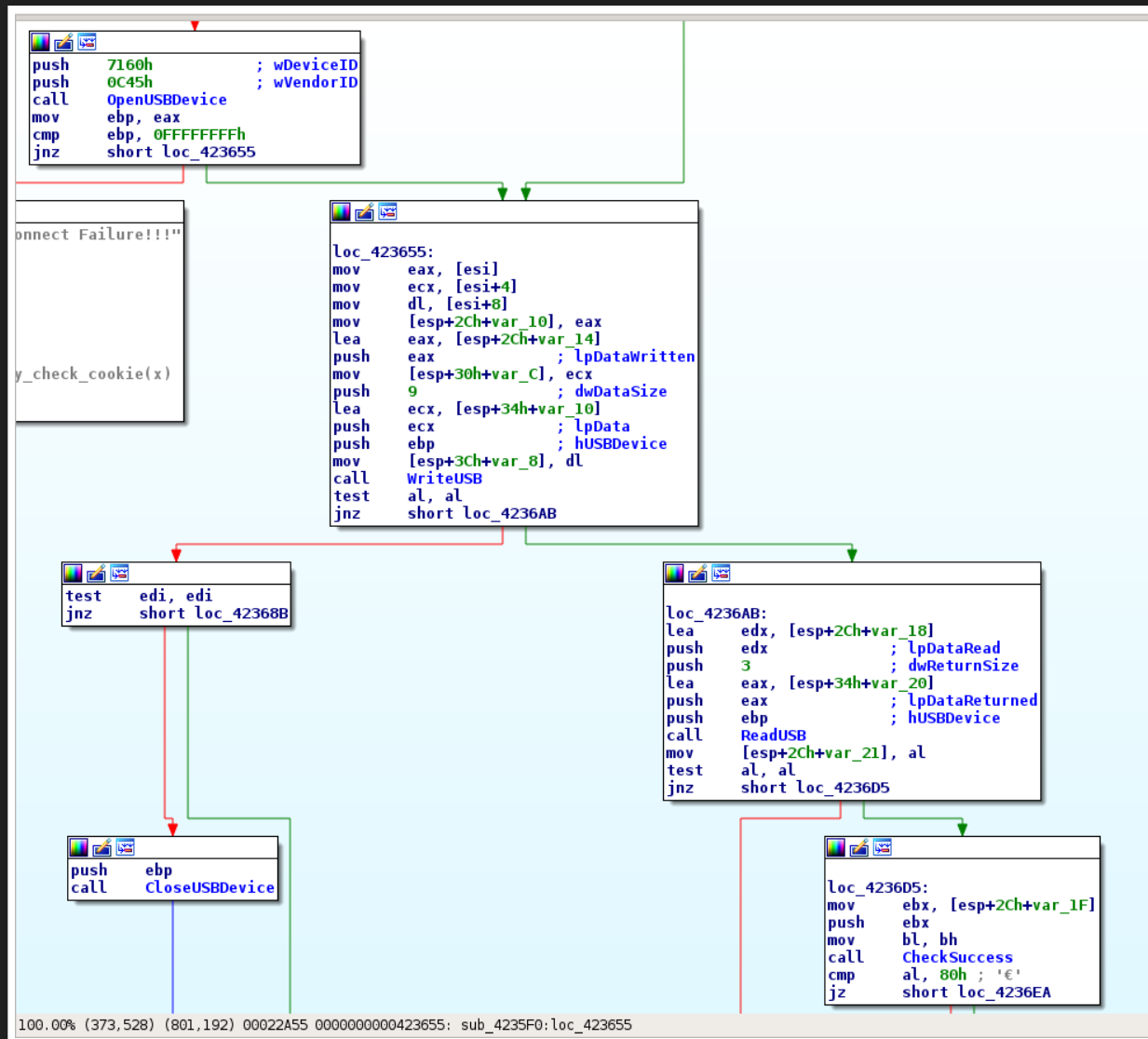
Straight to the interesting bit! 🤖

Directio	Typ	Address	Text
D...	r	sub_4235F0+86	call WriteUSB
D...	r	sub_41C4D0+2A2	cmp WriteUSB, 0
	w	sub_41C4D0+284	mov WriteUSB, eax

Line 1 of 3

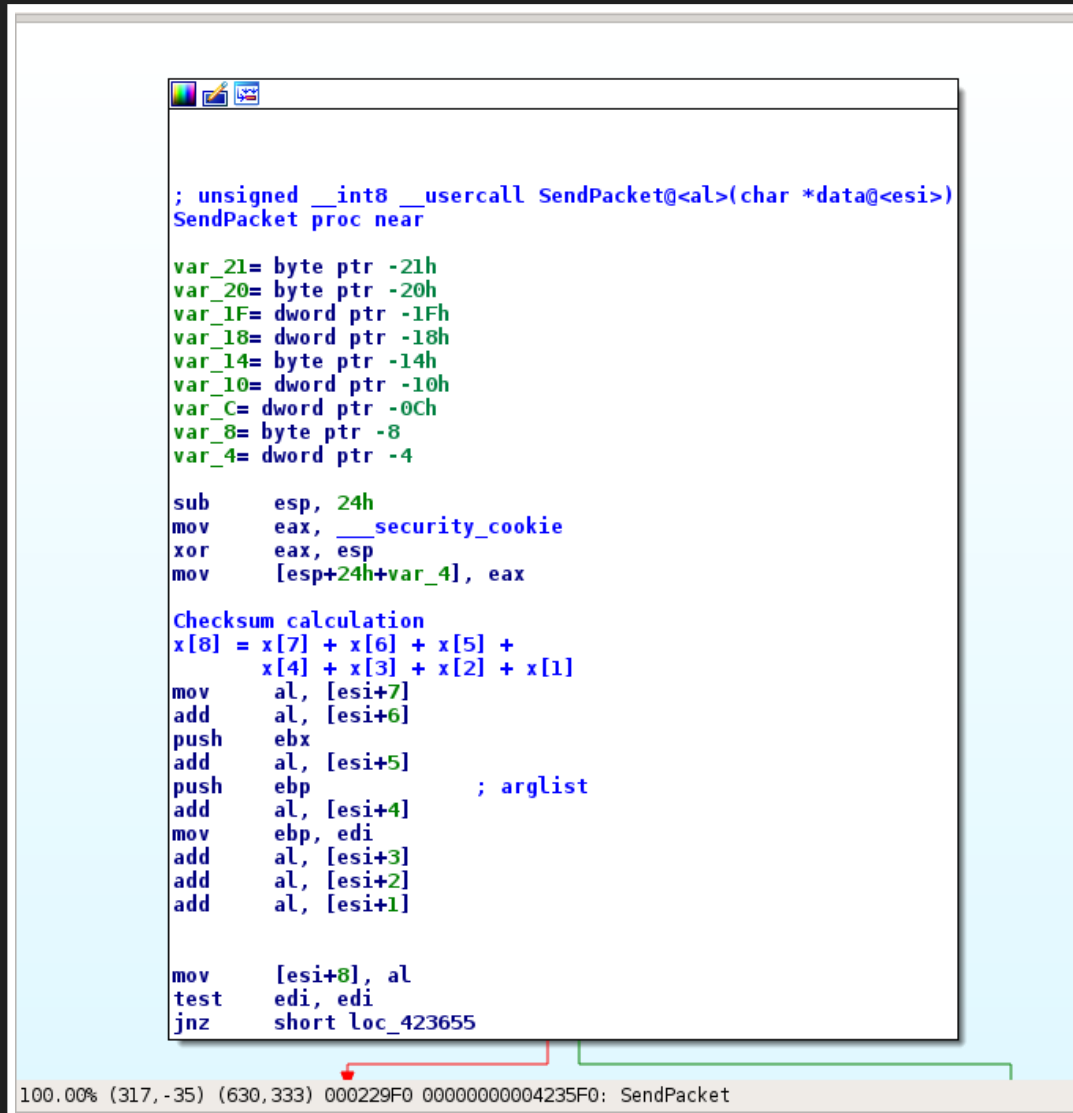
OK Cancel Search Help

One caller looks promising



Bingo 😊

Starting to learn about the protocol



```
; unsigned __int8 __usercall SendPacket@<al>(char *data@<esi>)
SendPacket proc near

var_21= byte ptr -21h
var_20= byte ptr -20h
var_1F= dword ptr -1Fh
var_18= dword ptr -18h
var_14= byte ptr -14h
var_10= dword ptr -10h
var_C= dword ptr -0Ch
var_8= byte ptr -8
var_4= dword ptr -4

sub     esp, 24h
mov     eax, ___security_cookie
xor     eax, esp
mov     [esp+24h+var_4], eax

Checksum calculation
x[8] = x[7] + x[6] + x[5] +
      x[4] + x[3] + x[2] + x[1]
mov     al, [esi+7]
add     al, [esi+6]
push    ebx
add     al, [esi+5]
push    ebp           ; arglist
add     al, [esi+4]
mov     ebp, edi
add     al, [esi+3]
add     al, [esi+2]
add     al, [esi+1]

mov     [esi+8], al
test    edi, edi
jnz     short loc_423655
```

100.00% (317, -35) (630, 333) 000229F0 00000000004235F0: SendPacket

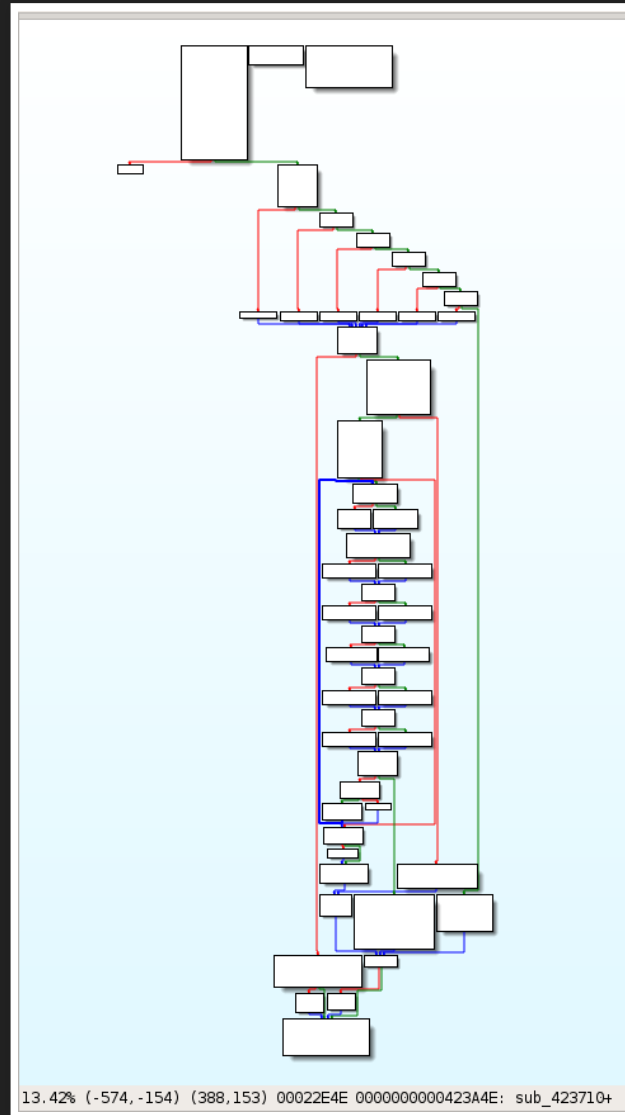
Only one function that calls into it 😊

Direction	Type	Address	Text
D...	p	sub_423710+18B	call SendPacket
D...	p	sub_423710+33E	call SendPacket

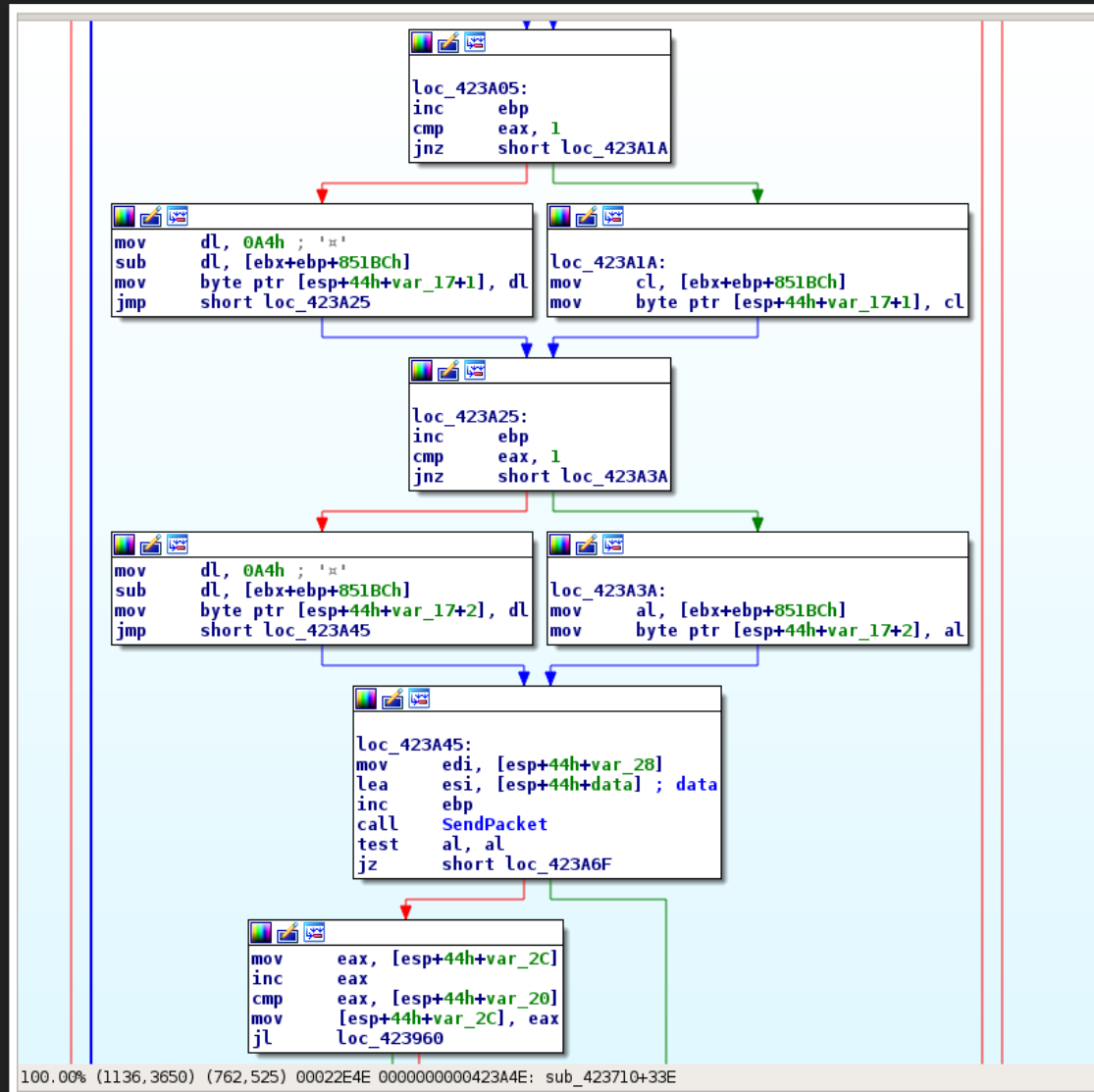
Line 2 of 2

OK Cancel Search Help

Big, but still learning!



Protocol seems to have an optional subtraction?



We also have the initial packet

```
jnz      Loc_42385E
Zero the initial packet
loc_42385E:
xor      eax, eax
mov      dword ptr [esp+44h+data+1], eax
mov      dword ptr [esp+44h+data+5], eax

data[0] = '\0'
mov      [esp+44h+data], al

data[3] = sizeish??? // Weird size log_2 thing
mov      al, [esp+44h+sizeish???]
mov      [esp+44h+data+3], al

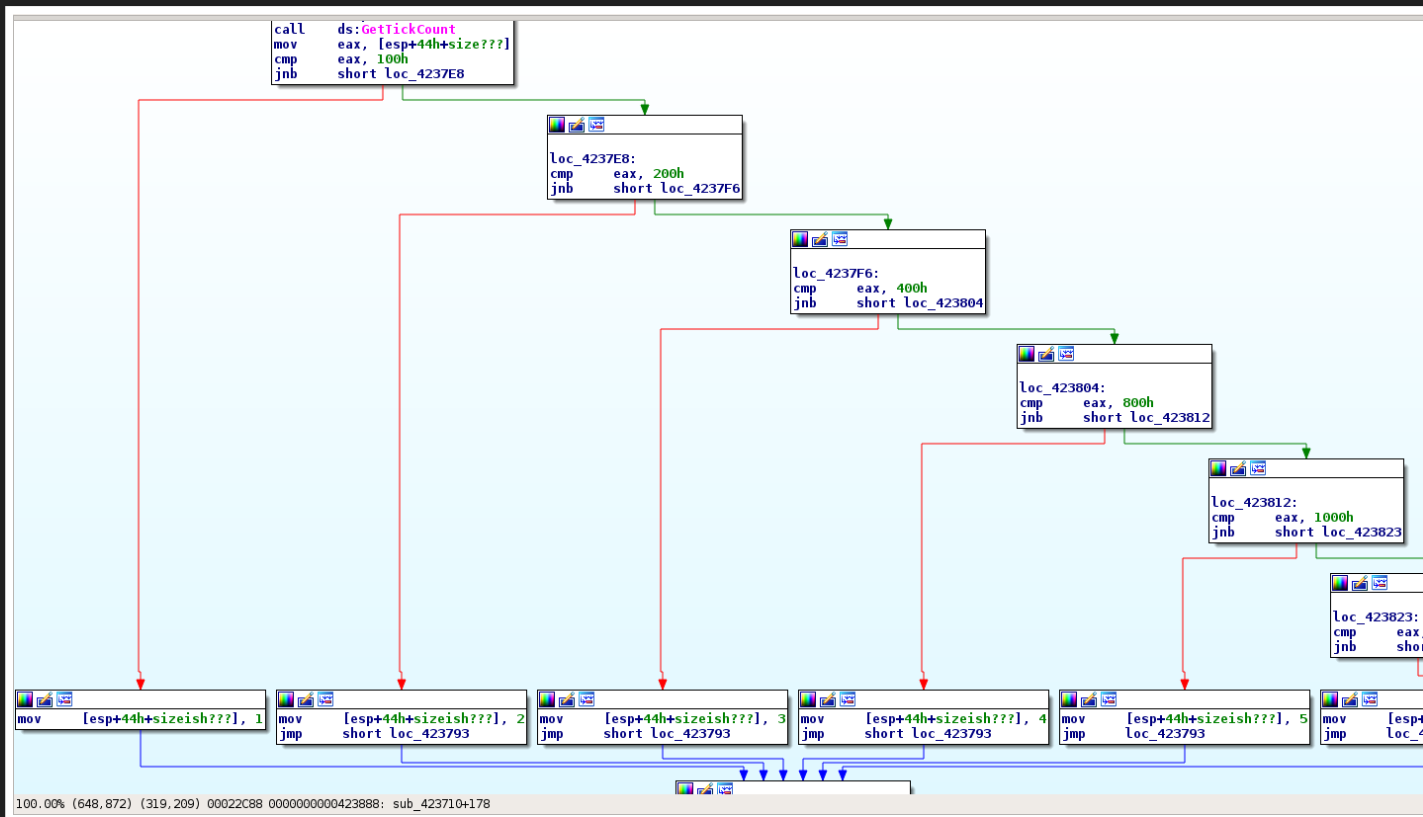
data[4] = (unsigned char)size??? // Looks like this is the size
mov      eax, [esp+44h+size???]
mov      [esp+44h+data+4], al
shr      eax, 8 ; size??? >> 8
lea      esi, [esp+44h+data] ; data

data[1] = '@'
mov      [esp+44h+data+1], 40h ; '@'
data[2] = '@'
mov      [esp+44h+data+2], 40h ; '@'
data[6] = '\0'
mov      [esp+44h+data+6], 0
data[7] = '\0'
mov      [esp+44h+data+7], 0

data[5] = (unsigned char)(size??? >> 8)
mov      [esp+44h+data+5], al


data = [\x00][\x40][\x40][sizeish???][size???.size???][\x00][\x00]
                                         ^^little endian^^
call     SendPacket
test     al, al
jnz      short loc_4238C5
```

100.00% (1320,1786) (672,375) 00022C88 0000000000423888: sub_423710+178



$$\begin{aligned}
 sizeish &= \log_2 \left(\frac{size}{64} \right) \\
 &= \log_2(size) - 6
 \end{aligned}$$

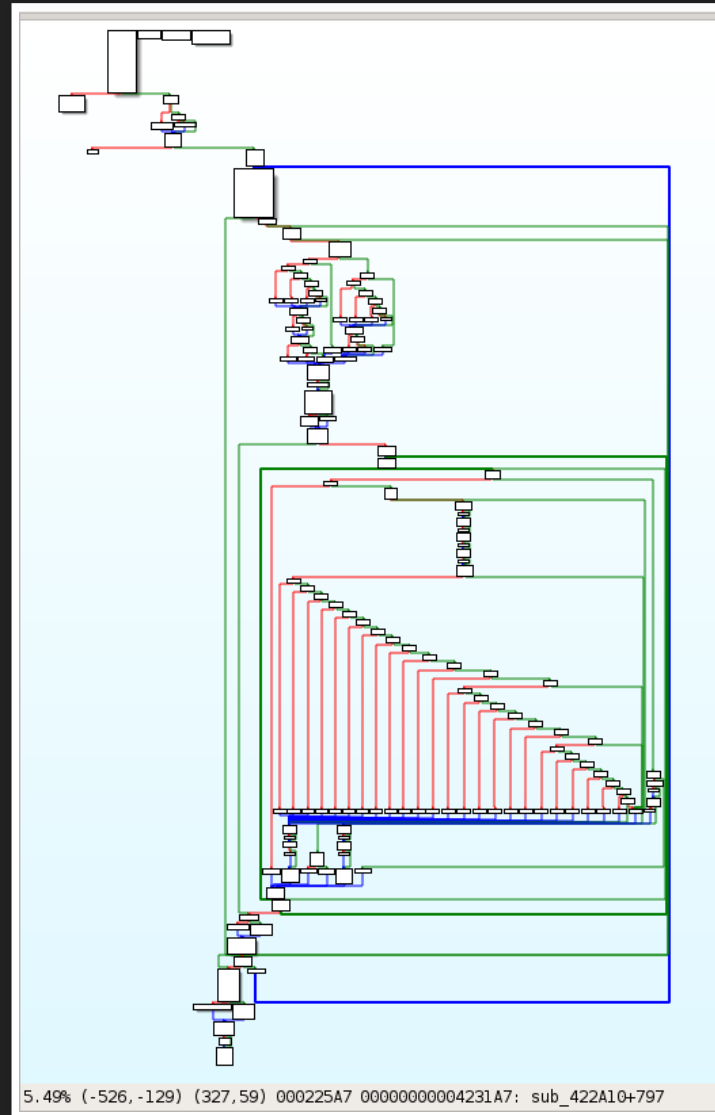
Also only one caller!

Directio	Typ	Address	Text
	Up	p sub_422A10+797	call SendMessage

Line 1 of 1

OK Cancel Search Help

But it's bigger, and uglier



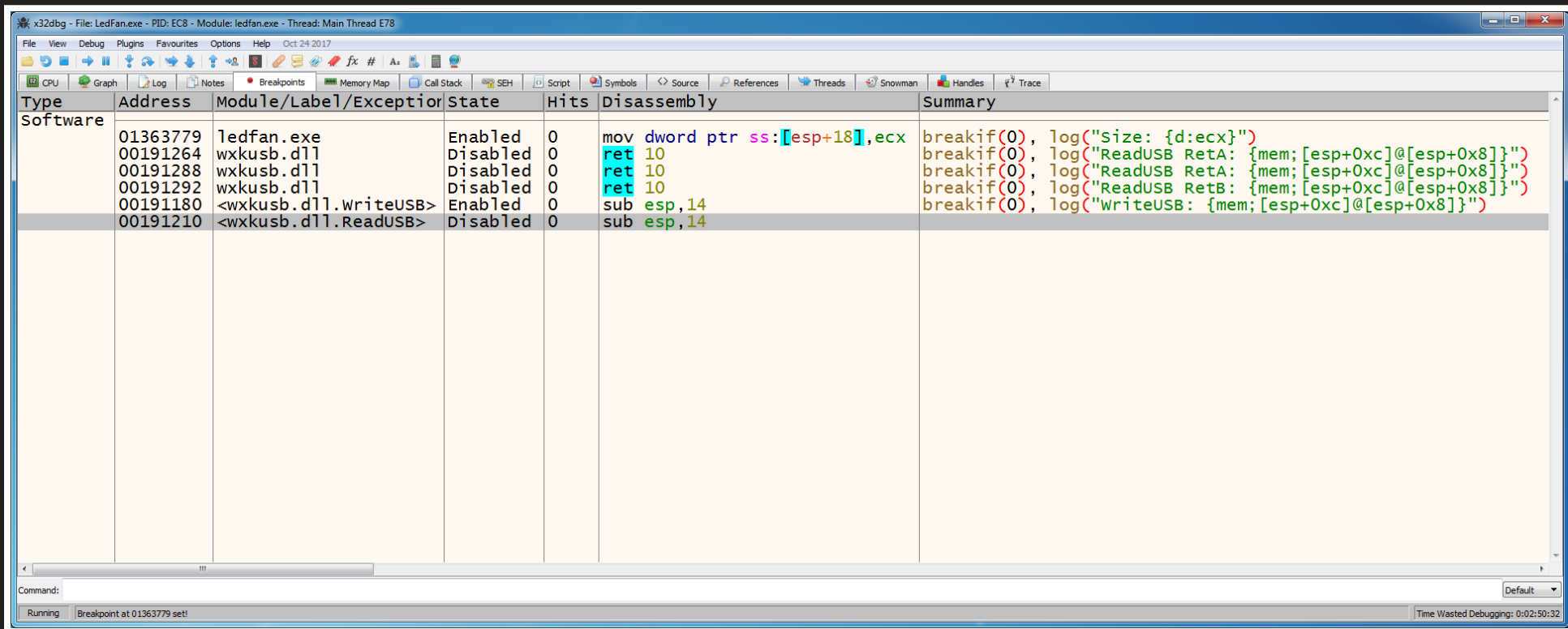
STEP BACK

- Where are we?
 - A `virtual` function
 - A `switch` statement nested in two loops
- How is the function called?
 - Can't find any obvious call into its vtable offset
- How are the loops and switches controlled?
 - Members of the `this` object
 - Window messages

Time for another plan 🤔

Remember that dynamic analysis?

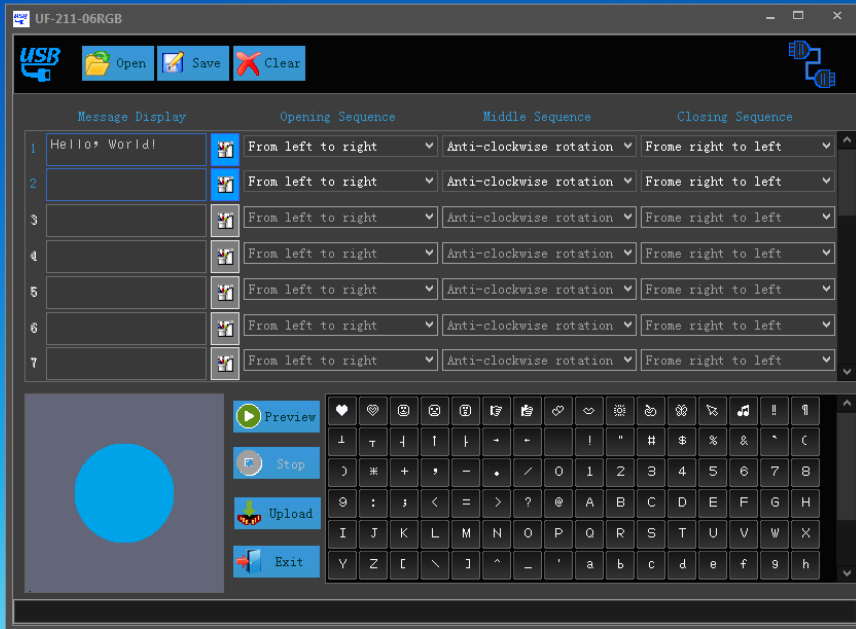
Lets validate some assumptions and dump the messages!



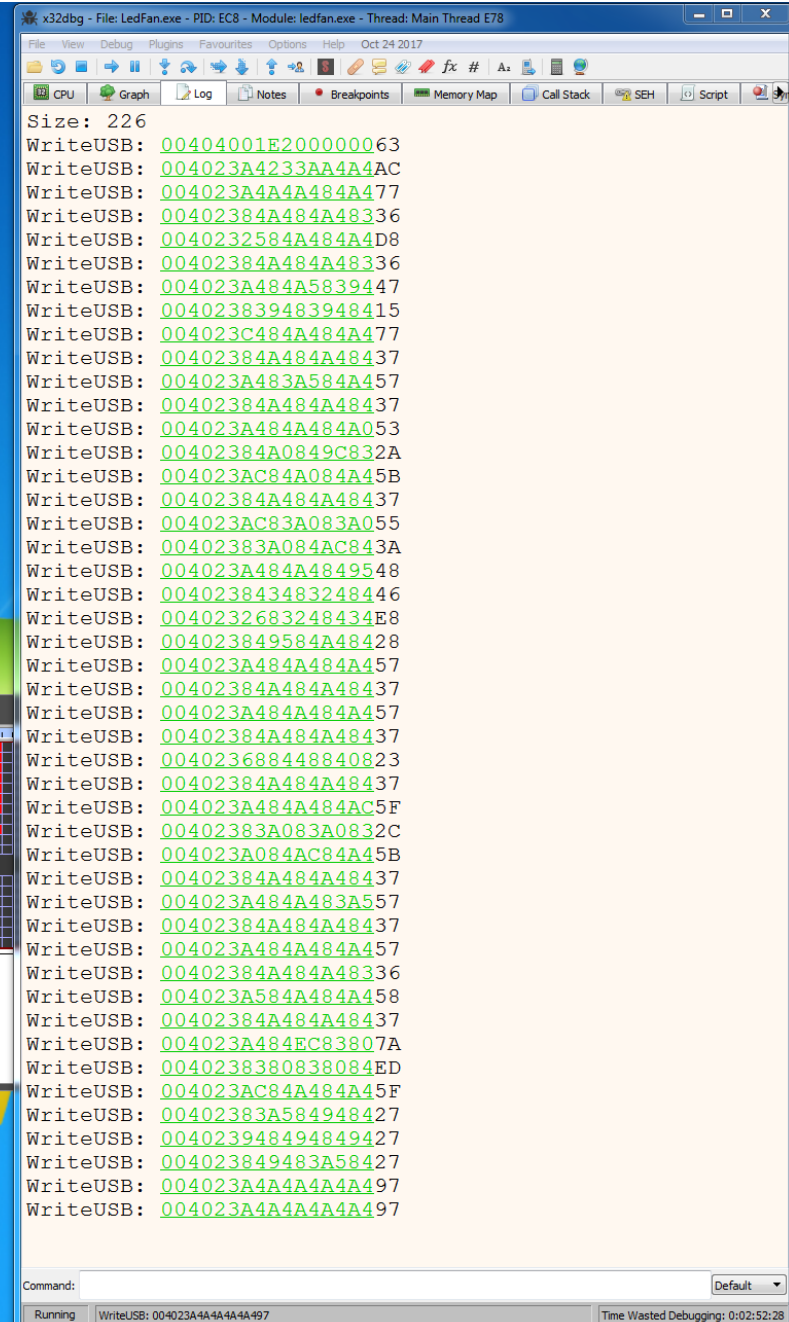
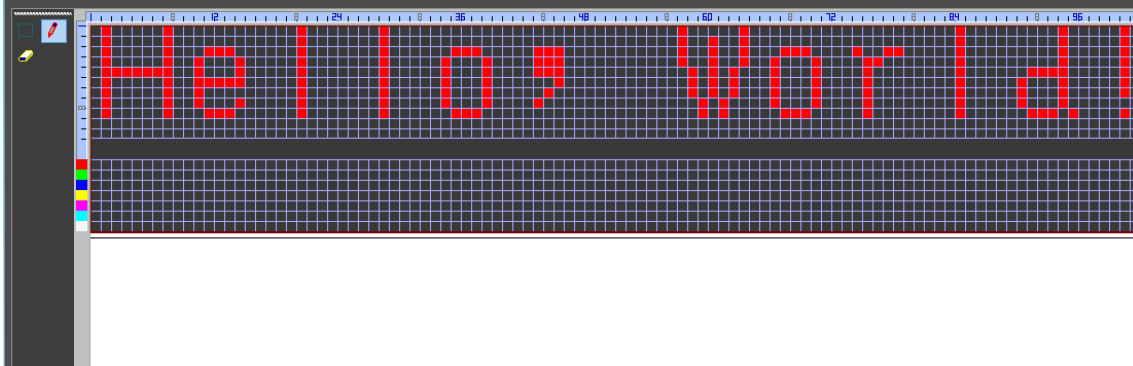
The screenshot shows the x32dbg debugger interface. The main window displays a list of breakpoints for the process 'ledfan.exe' (PID: EC8) and its module 'wxkusb.dll'. The breakpoints are listed in a table with columns: Type, Address, Module/Label/Exception, State, Hits, Disassembly, and Summary.

Type	Address	Module/Label/Exception	State	Hits	Disassembly	Summary
Software	01363779	ledfan.exe	Enabled	0	mov dword ptr ss:[esp+18],ecx	breakif(0), log("Size: {d:ecx}")
	00191264	wxkusb.dll	Disabled	0	ret 10	breakif(0), log("ReadUSB RetA: {mem; [esp+0xc]@[esp+0x8]}")
	00191288	wxkusb.dll	Disabled	0	ret 10	breakif(0), log("ReadUSB RetA: {mem; [esp+0xc]@[esp+0x8]}")
	00191292	wxkusb.dll	Disabled	0	ret 10	breakif(0), log("ReadUSB RetB: {mem; [esp+0xc]@[esp+0x8]}")
	00191180	<wxkusb.dll.WriteUSB>	Enabled	0	sub esp,14	breakif(0), log("writeUSB: {mem; [esp+0xc]@[esp+0x8]}")
	00191210	<wxkusb.dll.ReadUSB>	Disabled	0	sub esp,14	

The status bar at the bottom indicates the debugger is running and a breakpoint has been set at address 01363779. The time wasted debugging is 0:02:50:32.



Sect1 [144x20] - [104x20]



Single Red Dot

```
0040400122000000A3  
004023A4239AA4A40C  
004023A4A4A484A477  
00402384A484A48437  
004023A484A484A457  
00402384A484A3A456  
004023A4A4A4A4A497
```

Initial Packet

```
0040400122000000A3  
004023A4239AA4A40C  
004023A4A4A484A477  
00402384A484A48437  
004023A484A484A457  
00402384A484A3A456  
004023A4A4A4A4A497
```

Checksum

```
0040400122000000A3
004023A4239AA4A40C
004023A4A4A484A477
00402384A484A48437
004023A484A484A457
00402384A484A3A456
004023A4A4A4A4A497
```

$$40 + 40 + 01 + 22 + 00 + 00 + 00 = A3 \text{ 🍷}$$

Packet Start

```
0040400122000000A3  
004023A4239AA4A40C  
004023A4A4A484A477  
00402384A484A48437  
004023A484A484A457  
00402384A484A3A456  
004023A4A4A4A4A497
```

Remove the subtraction

```
00810A0000
0000002000
2000200020
0020002000
2000200100
0000000000
```

Eight columns

```
00810A0000
0000002000
2000200020
0020002000
2000200100
0000000000
```


Single pixel on

```
00810A0000
0000002000
2000200020
0020002000
2000200100
0000000000
```

Two Red Dots

```
0040400122000000A3
004023A4239AA4A40C
004023A4A4A484A477
00402384A484A48437
004023A484A484A457
00402384A384A3A455
004023A4A4A4A4A497
```

Remove the subtraction

```
00810A0000
0000002000
2000200020
0020002000
2001200100
0000000000
```

Two pixels on

```
00810A0000
0000002000
2000200020
0020002000
2001200100
0000000000
```

Full Vertical Red

0040400122000000A3
004023A4239AA4A40C
004023A4A4A484A477
00402384A484A48437
004023A484A484A457
00402384A47DA5A451
004023A4A4A4A4A497

Eleven pixels on

```
00810A0000
0000002000
2000200020
0020002000
200027FF00
0000000000
```

Full Vertical Blue

0040400122000000A3

004023A4239AA4A40C

004023A4A4A484A477

00402384A484A48437

004023A484A484A457

00402384A45DA5A431

004023A4A4A4A4A497

Eleven pixels on: blue

```
00810A0000  
0000002000  
2000200020  
0020002000  
200047FF00  
0000000000
```


Full Vertical Green

0040400122000000A3
004023A4239AA4A40C
004023A4A4A484A477
00402384A484A48437
004023A484A484A457
00402384A41DA5A4F1
004023A4A4A4A4A497

Eleven pixels on: green

```
00810A0000
0000002000
2000200020
0020002000
200087FF00
0000000000
```

But wait, the fan can do more!

- Multiple "pages" of messages
- As well as transition effects

But that's enough of the protocol analysis 😎

PROTOCOL REIMPLEMENTATION

With these results, reimplementation is very easy

- Python 3.6: `IntEnum`, f-strings, underscores in numeric literals
- `hidapi` package providing the abstraction for the "driverless" APIs



<https://git.io/fxawk>

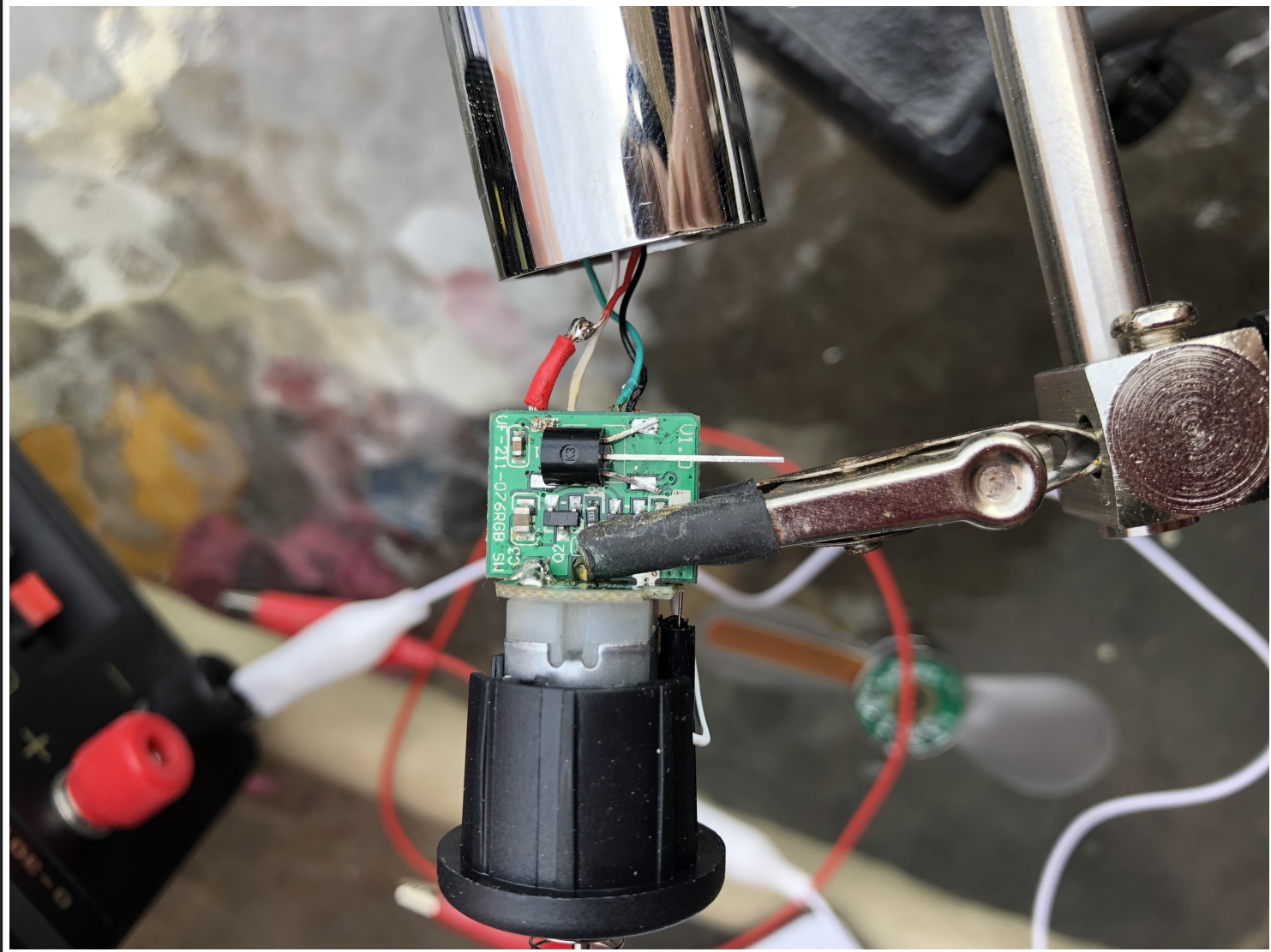
DRAWBACKS

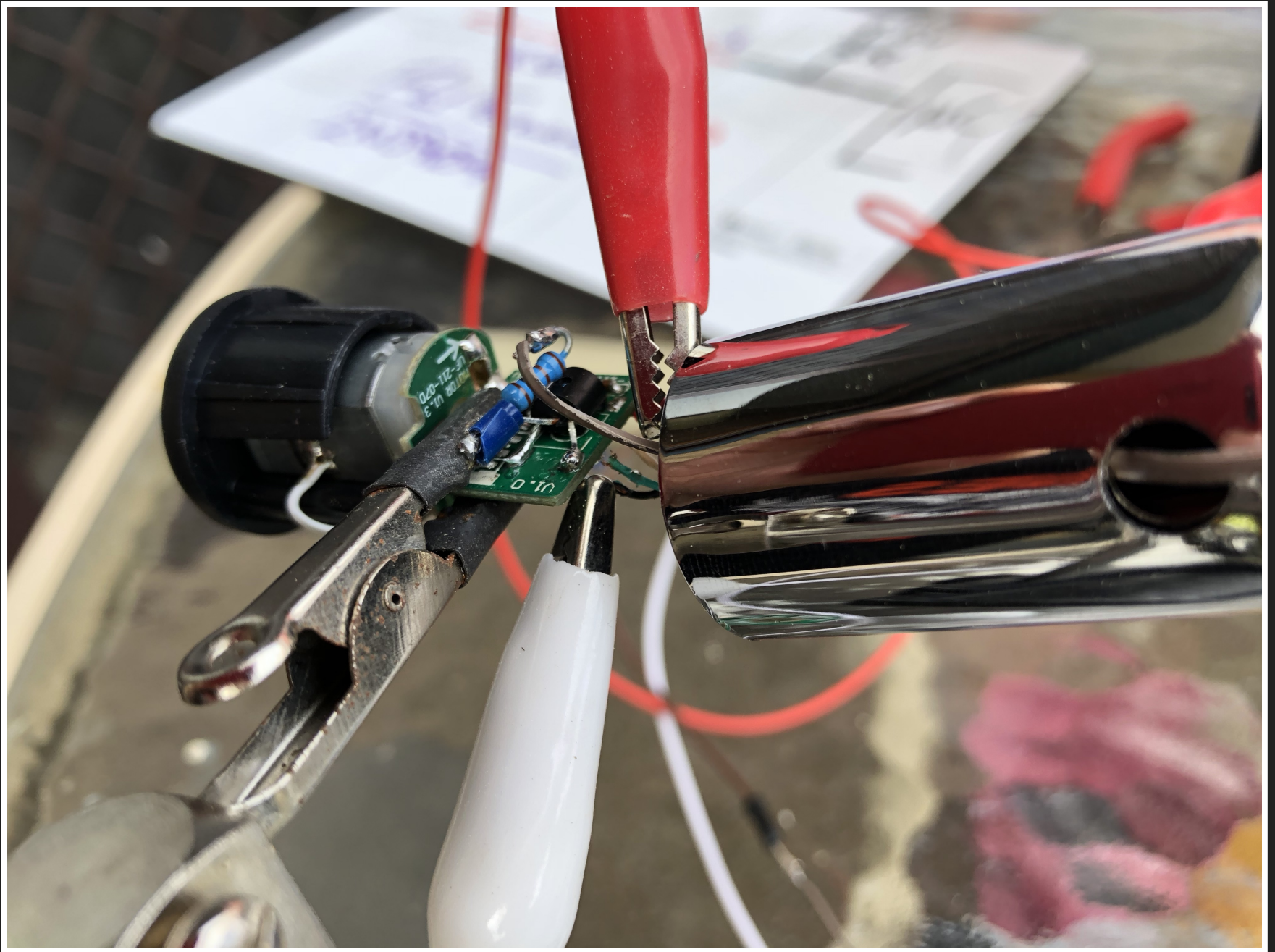
My main aim for the fan was to have notifications for home events

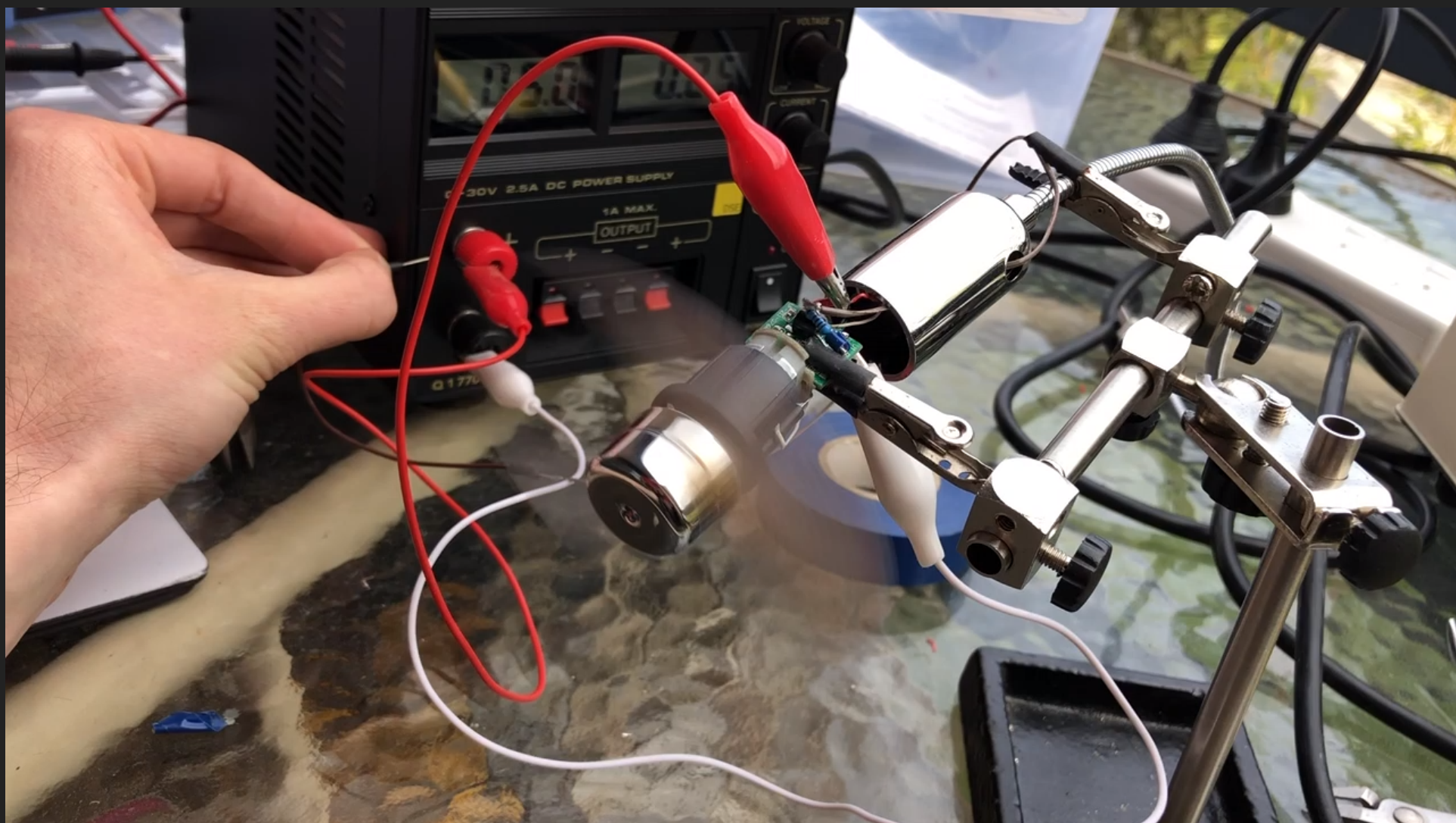
The fan's programming can only happen when it isn't spinning

There's no programmatic way of turning the motor off

Swap the button for a BJT 🧐

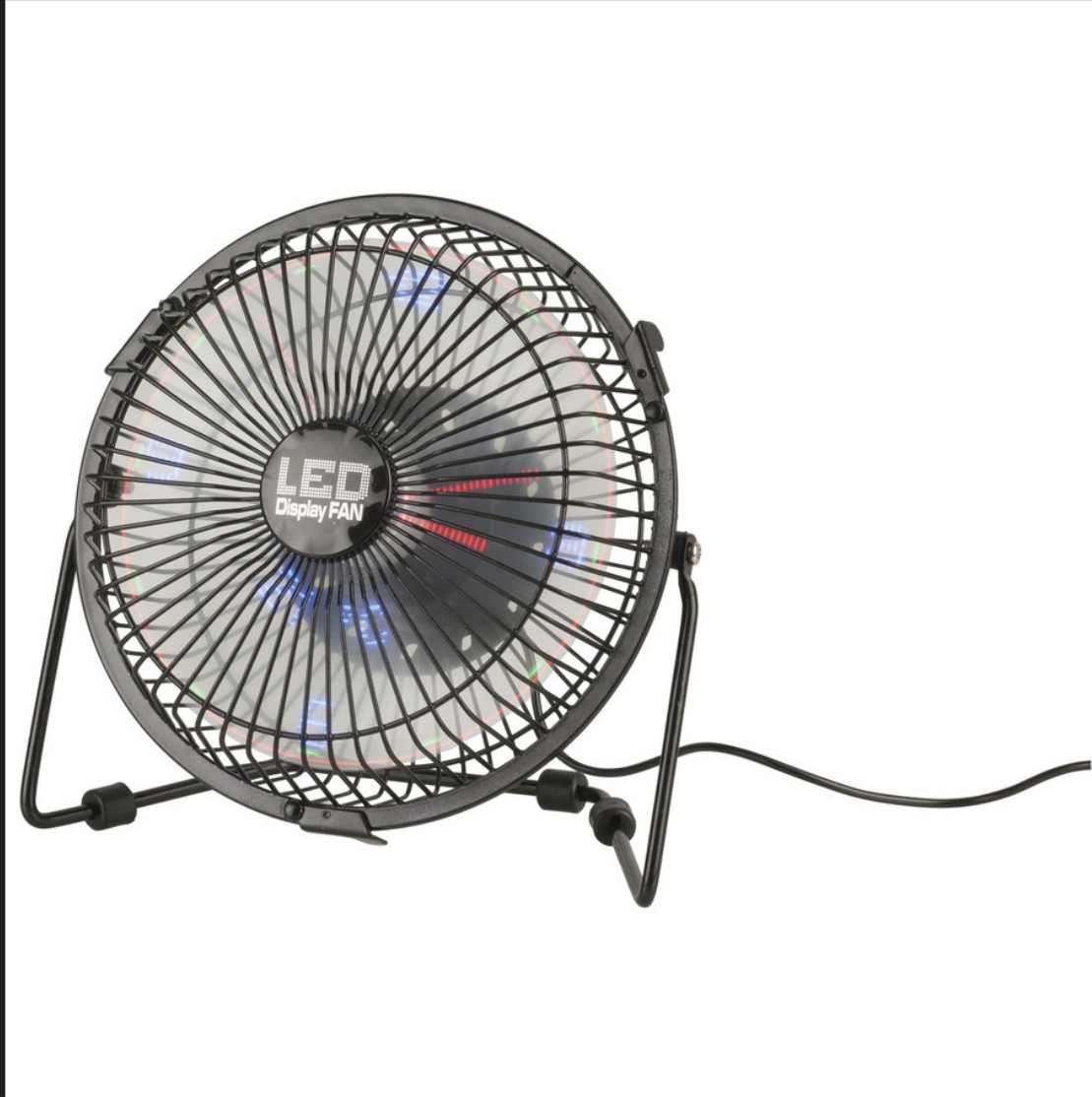






But it shorted when I put the case back on 😞

I have a new fan now, it's looking much better 😊



QUESTIONS

